SVD and Higher Order Correlations for Distributed Data

Ravi Kannan, Santosh Vempala, David Woodruff

June 15, 2014
Distributed Matrices

- Input Matrix A distributed among s servers.
Distributed Matrices

- Input Matrix $A$ distributed among $s$ servers.
- Example: Customer-Product matrix
Distributed Matrices

- Input Matrix A distributed among s servers.
- Example: Customer-Product matrix
  - For $t = 1, 2, \ldots, s$, Bundle of goods bought by customers from the $t$th shop is stored in server $t$. Call server $t$’s matrix $A^t$. 
Distributed Matrices

- Input Matrix $A$ distributed among $s$ servers.
- Example: Customer-Product matrix
  - For $t = 1, 2, \ldots, s$, Bundle of goods bought by customers from the t th shop is stored in server t. Call server t ’s matrix $A^t$.
  - Customer-Product Matrix $A = A^1 + A^2 + \cdots + A^t$. 
Distributed Matrices

- Input Matrix $A$ distributed among $s$ servers.
- Example: Customer-Product matrix
  - For $t = 1, 2, \ldots, s$, Bundle of goods bought by customers from the $t$th shop is stored in server $t$. Call server $t$'s matrix $A^t$.
  - Customer-Product Matrix $A = A^1 + A^2 + \cdots + A^t$.
  - More general than the row-partition model in which each customer shops in only one shop.
The Low Rank Approximation (LRA) Problem

▶ Input: \( n \times d \) matrix \( A \) stored on \( s \) servers.

\[ A = A_1 + A_2 + \cdots + A_s. \]

We assume \( n \geq d \). Tall Skinny matrix.

▶ Output: Server \( t \) has \( n \times d \) matrix \( C_t \) satisfying

\[ C_t = \sum_{s=1}^{s} C_t \] has rank at most \( k; \)

\[ ||A - C||_F \leq (1 + \varepsilon)||A - C^*||_F, \] \( C^* \) = Best Possible Rank \( k \) approx to \( A \).

▶ Resources: Each Server is Poly time, Linear Space.

Communication in \( O(1) \) rounds. Bound the total number of words communicated.

▶ Main Theorems of this part: Near-Optimal Communication Algorithm : \( \Theta^*(skd) \). No \( n! \).
The Low Rank Approximation (LRA) Problem

- **Input**: $n \times d$ matrix $A$ stored on $s$ servers.
  - Server $t$ has an $n \times d$ matrix $A_t$.

- Output: Server $t$ has $n \times d$ matrix $C_t$ satisfying
  - $C_t$ has rank at most $k$;
  - $\|A - C_t\|_F \leq (1 + \varepsilon)\|A - C^*\|_F$, $C^*$ = Best Possible Rank $k$ approximation to $A$.

- Resources: Each Server is Poly time, Linear Space.
  - Communication in $O\left(\frac{1}{\varepsilon}\right)$ rounds. Bound the total number of words communicated.

- Main Theorems of this part: Near-Optimal Communication Algorithm: $\Theta^*(skd)$. No $n!$
The Low Rank Approximation (LRA) Problem

- **Input:** $n \times d$ matrix $A$ stored on $s$ servers.
  - Server $t$ has an $n \times d$ matrix $A^t$.
  - $A = A^1 + A^2 + \cdots + A^s$.

- **Output:** Server $t$ has $n \times d$ matrix $C^t$ satisfying $C = \sum_{t=1}^s C^t$ has rank at most $k$;
  - $\| A - C \|_F \leq (1 + \varepsilon) \| A - C^* \|_F$, $C^*$ = Best Possible Rank $k$ approximation to $A$.

- **Resources:** Each Server is Poly time, Linear Space.
  - Communication in $O\left(\frac{1}{\varepsilon}\right)$ rounds. Bound the total number of words communicated.

- **Main Theorems of this part:** Near-Optimal Communication Algorithm: $\Theta^*\left(\frac{s}{k}d\right)$. No $n!$
The Low Rank Approximation (LRA) Problem

- **Input**: \( n \times d \) matrix \( A \) stored on \( s \) servers.
  - Server \( t \) has an \( n \times d \) matrix \( A^t \).
  - \( A = A^1 + A^2 + \cdots + A^s \).
  - We assume \( n \geq d \). Tall Skinny matrix.
The Low Rank Approximation (LRA) Problem

- Input: \( n \times d \) matrix \( A \) stored on \( s \) servers.
  - Server \( t \) has an \( n \times d \) matrix \( A^t \).
  - \( A = A^1 + A^2 + \cdots + A^s \).
  - We assume \( n \geq d \). Tall Skinny matrix.

- Output: Server \( t \) has \( n \times d \) matrix \( C^t \) satisfying

\[
\| A - C^t \|_F \leq (1 + \epsilon) \| A - C^* \|_F,
\]
where \( C^* \) is the Best Possible Rank \( k \) approximation to \( A \).

- Resources: Each Server is Poly time, Linear Space. Communication in \( O(1) \) rounds. Bound the total number of words communicated.

- Main Theorems of this part: Near-Optimal Communication Algorithm: \( \Theta^* (s k d) \). No \( n! \)
The Low Rank Approximation (LRA) Problem

- Input: $n \times d$ matrix $A$ stored on $s$ servers.
  - Server $t$ has an $n \times d$ matrix $A^t$.
  - $A = A^1 + A^2 + \cdots + A^s$.
  - We assume $n \geq d$. Tall Skinny matrix.

- Output: Server $t$ has $n \times d$ matrix $C^t$ satisfying
  - $C = \sum_{t=1}^{s} C^t$ has rank at most $k$;
The Low Rank Approximation (LRA) Problem

- **Input:** \( n \times d \) matrix \( A \) stored on \( s \) servers.
  - Server \( t \) has an \( n \times d \) matrix \( A^t \).
  - \( A = A^1 + A^2 + \cdots + A^s \).
  - We assume \( n \geq d \). Tall Skinny matrix.

- **Output:** Server \( t \) has \( n \times d \) matrix \( C^t \) satisfying
  - \( C = \sum_{t=1}^{s} C^t \) has rank at most \( k \);
  - \( \|A - C\|_F \leq (1 + \varepsilon)\|A - C^*\|_F \), \( C^* \) = Best Possible Rank \( k \) approx to \( A \).
The Low Rank Approximation (LRA) Problem

- **Input:** $n \times d$ matrix $A$ stored on $s$ servers.
  - Server $t$ has an $n \times d$ matrix $A^t$.
  - $A = A^1 + A^2 + \cdots + A^s$.
  - We assume $n \geq d$. Tall Skinny matrix.

- **Output:** Server $t$ has $n \times d$ matrix $C^t$ satisfying
  - $C = \sum_{t=1}^{s} C^t$ has rank at most $k$;
  - $\|A - C\|_F \leq (1 + \varepsilon)\|A - C^*\|_F$, $C^* = $ Best Possible Rank $k$ approx to $A$.

- **Resources:** Each Server is Poly time, Linear Space. Communication in $O(1)$ rounds. Bound the total number of words communicated.
The Low Rank Approximation (LRA) Problem

- **Input:** \( n \times d \) matrix \( A \) stored on \( s \) servers.
  - Server \( t \) has an \( n \times d \) matrix \( A^t \).
  - \( A = A^1 + A^2 + \cdots + A^s \).
  - We assume \( n \geq d \). Tall Skinny matrix.

- **Output:** Server \( t \) has \( n \times d \) matrix \( C^t \) satisfying
  - \( C = \sum_{t=1}^{s} C^t \) has rank at most \( k \);
  - \( \|A - C\|_F \leq (1 + \varepsilon)\|A - C^*\|_F \), \( C^* = \) Best Possible Rank \( k \) approx to \( A \).

- **Resources:** Each Server is Poly time, Linear Space. Communication in \( O(1) \) rounds. Bound the total number of words communicated.

- **Main Theorems of this part:** Near-Optimal Communication Algorithm: \( \Theta^*(skd) \). No \( n! \)
Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. If $S$ is a $m \times n$ random sign matrix and $U$ is the projection onto row space of $SA$, then, the best rank $k$ approximation to $AU$ does the job.
Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. If $S$ is a $m \times n$ random sign matrix and $U$ is the projection onto row space of $SA$, then, the best rank $k$ approximation to $AU$ does the job.

- **Proposed Algorithm:**
  - Central Processor (CP) selects $S$ and sends it to each server.
  - Server $t$ finds $SA_t$, sends it to CP.
  - CP finds $SA = \sum_t SA_t$, finds $U$ and sends it all servers.
  - Server $t$ finds $A_tU$, sends it to CP. CP finds $AU$, does SVD. Done.

- Sending $S$: $k$-wise independence enough. Send only seed of pseudo-random $S$. Length $O^*(k)$.
- Sending $A_tU = B_t$ which is $n \times O^*(k)$. 
Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. If $S$ is a $m \times n$ random sign matrix and $U$ is the projection onto row space of $SA$, then, the best rank $k$ approximation to $AU$ does the job.

- **Proposed Algorithm:**
  - Central Processor (CP) selects $S$ and sends it to each server.
  - Server $t$ finds $SA^t$, sends it to CP.
  - Server $t$ finds $A^tU$, sends it to CP. CP finds $AU$, does SVD. Done.

- Sending $S$: $k$-wise independence enough. Send only seed of pseudo-random $S$. Length $O^{*}(k)$. Sending $A^tU = B^t$ which is $n \times O^{*}(k)$. 
Clarkson and Woodruff $m \in O^*(k)$. If $S$ is a $m \times n$ random sign matrix and $U$ is the projection onto row space of $SA$, then, the best rank $k$ approximation to $AU$ does the job.

Proposed Algorithm:
- Central Processor (CP) selects $S$ and sends it to each server.
- Server $t$ finds $SA^t$, sends it to CP.
- CP finds $SA = \sum_t SA^t$, finds $U$ and sends it all servers.
Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. If $S$ is a $m \times n$ random sign matrix and $U$ is the projection onto row space of $SA$, then, the best rank $k$ approximation to $AU$ does the job.

- **Proposed Algorithm:**
  - Central Processor (CP) selects $S$ and sends it to each server.
  - Server $t$ finds $SA^t$, sends it to CP.
  - CP finds $SA = \sum_t SA^t$, finds $U$ and sends it all servers.
  - Server $t$ finds $A^tU$, sends it to CP. CP finds $AU$, does SVD. Done.
Clarkson and Woodruff $m \in O^*(k)$. If $S$ is a $m \times n$ random sign matrix and $U$ is the projection onto row space of $SA$, then, the best rank $k$ approximation to $AU$ does the job.

Proposed Algorithm:

- Central Processor (CP) selects $S$ and sends it to each server.
- Server $t$ finds $SA^t$, sends it to CP.
- CP finds $SA = \sum_t SA^t$, finds $U$ and sends it all servers.
- Server $t$ finds $A^tU$, sends it to CP. CP finds $AU$, does SVD. Done.

Sending $S$: $k-$wise independence enough. Send only seed of pseudo-random $S$. Length $O^*(k)$. Sending $A^tU = B^t$ which is $n \times O^*(k)$. 

Problem Reduced to

- **Server t** has $n \times k$ matrix $B^t$. Do SVD of $B = \sum_t B^t$. 

- (Simple) Lemma: If $P$ is a pseudo-random $O(\ast) \times n$ matrix, for all $x \in \mathbb{R}^k$, $|PBx| \approx \varepsilon |Bx|$, so SVD of $PB$ suffices to solve low-rank approx (LRA) of $B$.

- "Suffices" does not immediately give LRA to $A$. See paper for full proof.

- This two-stage adaptive sketching process has since played an important role in several follow-up works like CUR Matrix Factorization by Boutsides and Woodruff.

- Lower Bound: If server $t$ has now $k \times d$ matrix $A^t$ and finally, we have to have LRA to $A = \sum A^t$ on one server, clearly need $\Omega(skd)$ communication. But here, we only need server $t$ to end up with $C^t$, so sum of $C^t$ is an LRA.

- Trick: Server 1 and 2 special-have $A_1 = I_d$ and $A_2 = -I_d$, so $A = \sum_{t \geq 3} A^t$. At end, server 1 has to learn the projection to the LRA subspace which gives away the actual sum.
Problem Reduced to

- Server $t$ has $n \times k$ matrix $B^t$. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If $P$ is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in \mathbb{R}^k$, $|PBx| \approx \varepsilon |Bx|$, so SVD of $PB$ suffices to solve low-rank approx (LRA) of $B$.

"Suffices" does not immediately give LRA to $A$. See paper for full proof.

This two-stage adaptive sketching process has since played an important role in several follow-up works like CUR Matrix Factorization by Boutsides and Woodruff.

Lower Bound: If server $t$ has now $k \times d$ matrix $A^t$ and finally, we have to have LRA to $A = \sum A^t$ on one server, clearly need $\Omega(skd)$ communication. But here, we only need server $t$ to end up with $C^t$, so sum of $C^t$ is an LRA.

Trick: Server 1 and 2 special - have $A^1 = I_d$ and $A^2 = -I_d$, so $A = \sum_{t \geq 3} A^t$. At end, server 1 has to learn the projection to the LRA subspace which gives away the actual sum.
Problem Reduced to

- Server $t$ has $n \times k$ matrix $B^t$. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If $P$ is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in \mathbb{R}^k$, $|PBx| \approx_\epsilon |Bx|$, so SVD of $PB$ suffices to solve low-rank approx (LRA) of $B$.
- “Suffices” does not immediately give LRA to $A$. See paper for full proof.
Problem Reduced to

- Server $t$ has $n \times k$ matrix $B^t$. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If $P$ is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in \mathbb{R}^k$, $|PBx| \approx _\varepsilon |Bx|$, so SVD of $PB$ suffices to solve low-rank approx (LRA) of $B$.
- “Suffices” does not immediately give LRA to $A$. See paper for full proof.
- This two-stage adaptive sketching process has since played an important role in several follow-up works like CUR Matrix Factorization by Boutsides and Woodruff.
Problem Reduced to

- Server $t$ has $n \times k$ matrix $B^t$. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If $P$ is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in \mathbb{R}^k$, $|PBx| \approx_{\varepsilon} |Bx|$, so SVD of $PB$ suffices to solve low-rank approx (LRA) of $B$.
- “Suffices” does not immediately give LRA to $A$. See paper for full proof.
- This two-stage adaptive sketching process has since played an important role in several follow-up works like CUR Matrix Factorization by Boutsides and Woodruff.
- Lower Bound: If server $t$ has now $k \times d$ matrix $A^t$ and finally, we have to have LRA to $A = \sum_t A^t$ on one server, clearly need $\Omega(skd)$ communication. But here, we only need server $t$ to end up with $C^t$, so sum of $C^t$ is an LRA.
Problem Reduced to

- Server \( t \) has \( n \times k \) matrix \( B^t \). Do SVD of \( B = \sum_t B^t \).
- (Simple) Lemma: If \( P \) is a pseudo-random \( O^*(k) \times n \) matrix, for all \( x \in \mathbb{R}^k \), \(|PBx| \approx_\varepsilon |Bx|\), so SVD of \( PB \) suffices to solve low-rank approx (LRA) of \( B \).
- “Suffices” does not immediately give LRA to \( A \). See paper for full proof.
- This two-stage adaptive sketching process has since played an important role in several follow-up works like - CUR Matrix Factorization by Boutsides and Woodruff.
- Lower Bound: If server \( t \) has now \( k \times d \) matrix \( A^t \) and finally, we have to have LRA to \( A = \sum_t A^t \) on one server, clearly need \( \Omega(skd) \) communication. But here, we only need server \( t \) to end up with \( C^t \), so sum of \( C^t \) is an LRA.
- Trick: Server 1 and 2 special- have \( A^1 = I_d \) and \( A^2 = -I_d \), so \( A = \sum_{t \geq 3} A^t \). At end, server 1 has to learn the projection to the LRA subspace which gives away the actual sum.
Interest in Distributed Optimization in ML.
Other Such Problems not in the paper

- Interest in Distributed Optimization in ML.
- Provably: For Linear Programming \( Ax \leq b, A_{n \times d} \), with each constraint residing in one server, \( O^*(d^4) \) communication suffices, but \( \text{poly}(d) \) rounds. (Ellipsoid Algorithm).
Other Such Problems not in the paper

- Interest in Distributed Optimization in ML.
- Provably: For Linear Programming \( Ax \leq b, A_{n \times d}, \) with each constraint residing in one server, \( O^*(d^4) \) communication suffices, but \( \text{poly}(d) \) rounds. (Ellipsoid Algorithm).
Other Such Problems not in the paper

- Interest in Distributed Optimization in ML.
- Provably: For Linear Programming: $Ax \leq b$, $A_{n \times d}$, with each constraint residing in one server, $O^*(d^4)$ communication suffices, but $\text{poly}(d)$ rounds. (Ellipsoid Algorithm).
- Lower bound for Linear Equations: $Ax = b$: $\Omega(d^2)$ communication.
Other Such Problems not in the paper

- Interest in Distributed Optimization in ML.
- Provably: For Linear Programming: $Ax \leq b$, $A_{n \times d}$, with each constraint residing in one server, $O^*(d^4)$ communication suffices, but $\text{poly}(d)$ rounds. (Ellipsoid Algorithm).
- Lower bound for Linear Equations: $Ax = b$: $\Omega(d^2)$ communication.
- Linear and non-linear optimization, fault tolerant SVD, all of interest.
Higher Order Correlations-Examples

- Example: Time-series data for many events. Each event resides fully on one server.
Higher Order Correlations-Examples

- Example: Time-series data for many events. Each event resides fully on one server.
- Estimate to relative error $\varepsilon$ number of 7-tuples $(E_1, E_2, E_3, t_1, t_2, t_3, t_4)$ such that each of $E_1, E_2, E_3$ occurs at each of the times $t_1, t_2, t_3, t_4$. 

- Bipartite graph. Estimate number of $K_{3,4}$.
- More generally: Each customer resides wholly on one server. Estimate number of $(C_1, C_2, C_3, P_1, P_2, P_3, P_4)$ such that each of the 3 customers $C_1, C_2, C_3$ buys at least $x$ amount of at least 3 of the 4 products $P_1, P_2, P_3, P_4$.

- Customer $i$ turns into a $(n^4)$ component vector $v_i$ and answer is (essentially) $||\sum_i v_i||_3^3 = \text{Sum of cubes of components of the sum of vectors } v_i$.
- In streaming, lower bounds tell us that the $(n^4)$ makes it impossible with polylog communication. But in this distributed model, the lower bound does not apply and we can in fact "do it".
Higher Order Correlations-Examples

- Example: Time-series data for many events. Each event resides fully on one server.

- Estimate to relative error $\varepsilon$ number of 7-tuples $(E_1, E_2, E_3, t_1, t_2, t_3, t_4)$ such that each of $E_1, E_2, E_3$ occurs at each of the times $t_1, t_2, t_3, t_4$.

- Bipartite graph. Estimate number of $K_{3,4}$.
Higher Order Correlations-Examples

- Example: Time-series data for many events. Each event resides fully on one server.
- Estimate to relative error $\varepsilon$ number of 7-tuples $(E_1, E_2, E_3, t_1, t_2, t_3, t_4)$ such that each of $E_1, E_2, E_3$ occurs at each of the times $t_1, t_2, t_3, t_4$.
- Bipartite graph. Estimate number of $K_{3,4}$.
- More generally: Each customer resides wholly on one server. Estimate number of $(C_1, C_2, C_3, P_1, P_2, P_3, P_4)$ such that each of the 3 customers $C_1, C_2, C_3$ buys at least $x$ amount of at least 3 of the 4 products $P_1, P_2, P_3, P_4$. 

$\sum || \sum v_i ||^3 = \text{Sum of cubes of components of the sum of vectors } v_i$. 

In streaming, lower bounds tell us that the 3 above ($>2$) makes it impossible with polylog communication. But in this distributed model, the lower bound does not apply and we can in fact "do it".
Higher Order Correlations-Examples

- Example: Time-series data for many events. Each event resides fully on one server.
- Estimate to relative error $\varepsilon$ number of 7-tuples $(E_1, E_2, E_3, t_1, t_2, t_3, t_4)$ such that each of $E_1, E_2, E_3$ occurs at each of the times $t_1, t_2, t_3, t_4$.
- Bipartite graph. Estimate number of $K_{3,4}$.
- More generally: Each customer resides wholly on one server. Estimate number of $(C_1, C_2, C_3, P_1, P_2, P_3, P_4)$ such that each of the 3 customers $C_1, C_2, C_3$ buys at least $x$ amount of at least 3 of the 4 products $P_1, P_2, P_3, P_4$.
- Customer $i$ turns into a $\binom{n}{4}$ component vector $v_i$ and answer is (essentially) $\| \sum_i v_i \|_3^3 = \text{Sum of cubes of components of the sum of vectors } v_i$. 
Higher Order Correlations-Examples

- Example: Time-series data for many events. Each event resides fully on one server.
- Estimate to relative error $\varepsilon$ number of 7-tuples $(E_1, E_2, E_3, t_1, t_2, t_3, t_4)$ such that each of $E_1, E_2, E_3$ occurs at each of the times $t_1, t_2, t_3, t_4$.
- Bipartite graph. Estimate number of $K_{3,4}$.
- More generally: Each customer resides wholly on one server. Estimate number of $(C_1, C_2, C_3, P_1, P_2, P_3, P_4)$ such that each of the 3 customers $C_1, C_2, C_3$ buys at least $x$ amount of at least 3 of the 4 products $P_1, P_2, P_3, P_4$.
- Customer $i$ turns into a $\binom{n}{4}$ component vector $v_i$ and answer is (essentially) $\| \sum_i v_i \|_3^3 = \text{Sum of cubes of components of the sum of vectors } v_i$.
- In streaming, lower bounds tell us that the 3 above ($> 2$) makes it impossible with polylog communication. But in this distributed model, the lower bound does not apply and we can in fact “do it”.
Higher Order Correlations

- **General Set-up:** A set of $n$ vectors $v_1, v_2, \ldots$, each wholly residing in one server.
Higher Order Correlations

- General Set-up: A set of $n$ vectors $v_1, v_2, \ldots$, each wholly residing in one server.
- $k, r$ fixed positive integers. $g : \mathbb{R}^k_+ \rightarrow \mathbb{R}_+$, any monotone function.
Higher Order Correlations

- **General Set-up:** A set of $n$-vectors $v_1, v_2, \ldots$, each wholly residing in one server.

- $k, r$ fixed positive integers. $g : \mathbb{R}_+^k \to \mathbb{R}_+$, any monotone function.

- $\sum_{j_1, j_2, \ldots, j_k \in [n]} (\sum_i g(v_{i,j_1}, v_{i,j_2}, \ldots, v_{i,j_k}))^r$ can be estimated to relative error $\varepsilon > 0$ in linear space, poly time, $O(1)$ rounds and total communication $O^*(s^{r+2})$. 

Further, we need $\Omega(s^r - 1)$ and the gap of $s^3$ can be closed when $k = 1$. [Classic problem for streaming-

"frequency moments"]
Higher Order Correlations

- General Set-up: A set of $n - v$ vectors $v_1, v_2, \ldots$, each wholly residing in one server.
- $k, r$ fixed positive integers. $g : \mathbb{R}_+^k \rightarrow \mathbb{R}_+$, any monotone function.
- $\sum_{j_1, j_2, \ldots, j_k \in [n]} \left( \sum_i g(v_{i, j_1}, v_{i, j_2}, \ldots, v_{i, j_k}) \right)^r$ can be estimated to relative error $\varepsilon > 0$ in linear space, poly time, $O(1)$ rounds and total communication $O^*(s^{r+2})$.
- Further, we need $\Omega(s^{r-1})$ and the gap of $s^3$ can be closed when $k = 1$. [Classic problem for streaming-“frequency moments”]
Idea of Algorithm, Generalizations

- Server \( t \) has vector \( u^t \). Want \( \| \sum_t u^t \|_3^3 \).

- Enough to sample \( i \) with probabilities proportional to \( (\sum_t u^t_i)^3 \).

- Basic Idea: Server \( t \) samples \( i \) with probabilities proportional to \( (u^t_i)^3 \).

- Everyone sends sample to CP. Some careful rejection sampling.

- In final theorem, more general functions than \( r \) th power are dealt with.

- Open: Characterize all functions \( f, g \) which can be handled.
Idea of Algorithm, Generalizations

- Server $t$ has vector $u^t$. Want $\| \sum_t u^t \|^3$.
- Enough to sample $i$ with probabilities proportional to $(\sum_t u^t_i)^3$. 

In final theorem, more general functions than $r$ th power are dealt with.

Open: Characterize all functions $f$, $g$ which can be handled.
Server $t$ has vector $u^t$. Want $\| \sum_t u^t \|_3^3$.

Enough to sample $i$ with probabilities proportional to $(\sum_t u_i^t)^3$.

Basic Idea: Server $t$ samples $i$ with probabilities proportional to $(u_i^t)^3$. 
Idea of Algorithm, Generalizations

- Server t has vector $u^t$. Want $\| \sum_t u^t \|^3_3$.
- Enough to sample i with probabilities proportional to $(\sum_t u^t_i)^3$.
- Basic Idea: Server t samples i with probabilities proportional to $(u^t_i)^3$.
- Everyone sends sample to CP. Some careful rejection sampling.
Idea of Algorithm, Generalizations

- Server $t$ has vector $u^t$. Want $\|\sum_t u^t\|^3_3$.
- Enough to sample $i$ with probabilities proportional to $(\sum_t u^t_i)^3$.
- Basic Idea: Server $t$ samples $i$ with probabilities proportional to $(u^t_i)^3$.
- Everyone sends sample to CP. Some careful rejection sampling.
- In final theorem, more general functions than $r$ th power are dealt with.
Server $t$ has vector $u^t$. Want $\| \sum_t u^t \|^3_3$.

Enough to sample $i$ with probabilities proportional to $(\sum_t u^t_i)^3$.

Basic Idea: Server $t$ samples $i$ with probabilities proportional to $(u^t_i)^3$.

Everyone sends sample to CP. Some careful rejection sampling.

In final theorem, more general functions than $r$ th power are dealt with.

Open: Characterize all functions $f, g$ which can be handled.